Apports des outils numériques sur l'enseignement de l'automatique : ateliers à partir de notebook Jupyter – Atelier 2

Culture Sciences de l'Ingénieur

Javier OJEDA

Édité le 06/07/2020

école normale supérieure paris—saclay

Atelier 2 : Le dimensionnement automatique de correcteur P, PI et PID à partir de la méthode de Ziegler-Nichols en boucle fermée

Cette ressource est l'un des trois ateliers illustrant la ressource « *Apports des outils numériques sur l'enseignement de l'automatique : ateliers à partir de notebook Jupyter* ».

Résumé : L'évolution des outils numériques aussi bien logiciels que du matériel permet d'envisager l'enseignement de l'automatique de manière plus illustrative et participative. Illustrative, car des outils numériques, le plus souvent libres, sont disponibles pour les étudiants afin de mettre en image des notions parfois complexes de l'automatique et participative, car au travers d'ateliers réalisés par les étudiants, ceux-ci peuvent appréhender des notions principales ou annexes par leurs simulations et expérimentations. Par ailleurs, participative, par le fait que les outils numériques permettent des échanges enseignant-apprenant en ligne via un site web, des plateformes moodle, slack, etc. Dans cette ressource, nous allons nous intéresser aux outils numériques logiciels par le biais du langage Python et du notebook Jupyter. Pour cela nous allons décrire trois ateliers enseignant-apprenant réalisables en ligne et qui abordent trois notions de l'automatique : l'identification, le dimensionnement automatique d'un correcteur de type PID et le calcul des pôles en boucle fermée.

Ressource publiée sur Culture Sciences de l'Ingénieur : https://eduscol.education.fr/sti/si-ens-paris-saclay

Atelier_2

June 12, 2020

1 Atelier n°2 : Calcul automatique de correcteurs P, PI, PID

Auteur : Javier OJEDA
Date : 25/05/2020
Version : 0.1

1.1 Importation des packages nécessaires

[1]:
```
import control as ctl # Package control pour l'automatique
```

Lien pour le package control

[2]:
```
#Affichage avec la bibliothèque graphique intégrée à Notebook
%matplotlib inline
import matplotlib.pyplot as plt # Package pour le représentation graphique
plt.rcParams.update({'font.size': 18}) # Changer la taille de police par défaut
plt.rcParams.update({'lines.linewidth': 3}) # Changer la taille des lignes

import numpy as np # Package pour le calcul scientifique

import warnings
warnings.filterwarnings('ignore') # Pour retirer les warnings intempestifs
```

1.2 Méthodologie

Objectif : Cet atelier doit permettre de mettre en place un processus de calcul automatique d'un correcteur P, PI ou PID en utilisant la notion de marge de gain pour un système d'ordre supérieur ou égal à 3.

Moyen : Nous utiliserons la marge de gain qui définie le gain que l'on peut appliquer au système en boucle ouverte pour obtenir un gain de 1 et une marge de phase de 0°.

A partir de la connaissance de la marge de gain et de sa fréquence associée, nous appliquerons la méthode de Ziegler-Nichols en boucle fermée pour calculer les gains des correcteurs.

Méthode de Ziegler-Nichols :

Correcteur	K_p	T_i	T_d
P	$0.5K_c$		

Correcteur	K_p	T_i	T_d
PI	$0.45K_c$	$0.83T_c$	
PID	$0.6K_c$	$0.58T_c$	$0.125T_c$
Léger dépassement	$0.33K_c$	$0.5T_c$	$0.33T_c$

Avec K_c le gain critique correspondant à la marge de gain et T_c la période associée à la marge de gain.

Le correcteur P aura comme forme : $C(p) = K_p$

Le correcteur PI aura comme forme : $C(p) = K_p \left(1 + \frac{1}{T_i p}\right)$

Le correcteur PID aura comme forme : $C(p) = K_p \left(1 + \frac{1}{T_i p} + T_d p\right)$

1.3 Définition du système à étudier

1.3.1 Sous la forme d'une fonction de transfert

[3]:
```
# Système à étudier
K_0 = 1.02              # Gain statique
TAU = 0.68e-3           # Constante de temps du 1er ordre
M = 0.4                 # Amortissement du 2nd ordre
W0 = 1/150e-6           # Pulsation du 2nd ordre

##############################################################################
# Fonction de transfert en BO
#

NUM = [K_0] # Numérateur de la fonction de transfert

# Dénominateur de la fonction de transfert par ordre décroissant en p ou s
DENUM = [TAU/W0**2, 2*M/W0*TAU + 1/W0**2, TAU + 2*M/W0, 1]

SYS_TF = ctl.tf(NUM, DENUM) # Fonction de transfert en p
```

Diagramme de Bode

[4]:
```
plt.figure(1, [10, 10])
mag, phase, omega = ctl.bode(SYS_TF, margins = True)
plt.savefig("bode_margin_A1.png", dpi=300)
```

Gm = 4.36 □(at 7231.02 rad/s), Pm = 166.12 deg (at 306.33 rad/s)

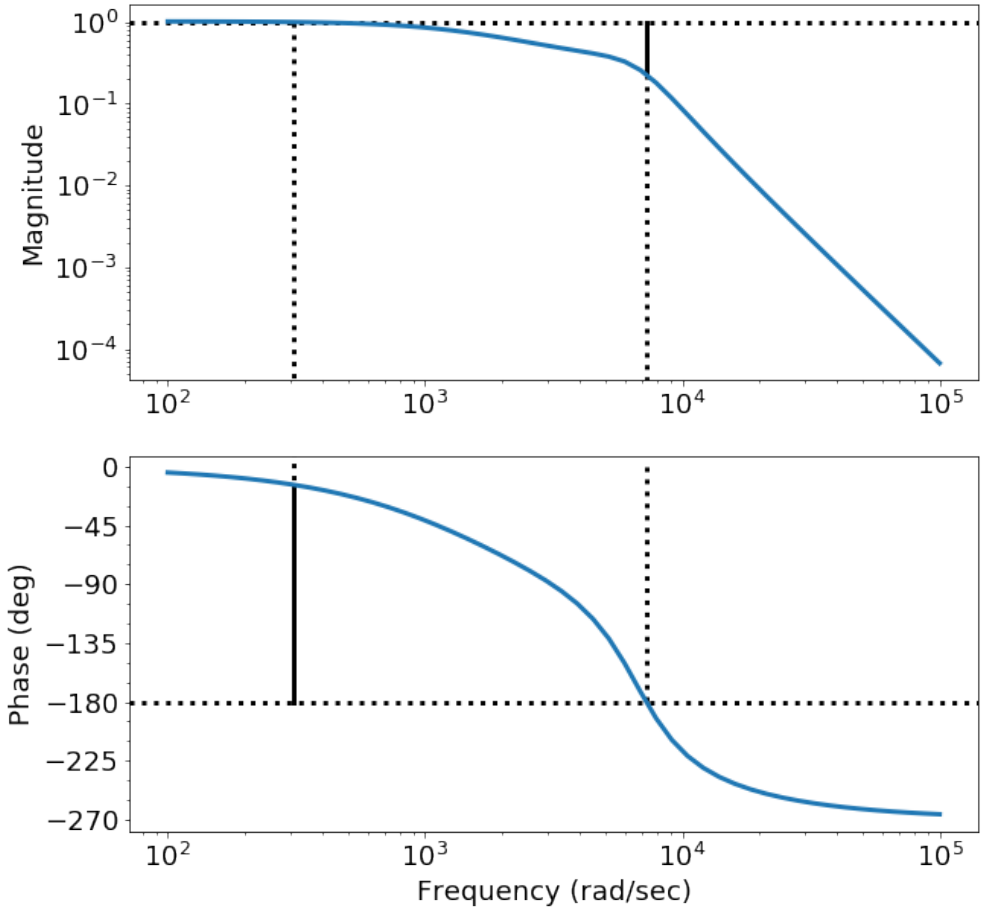
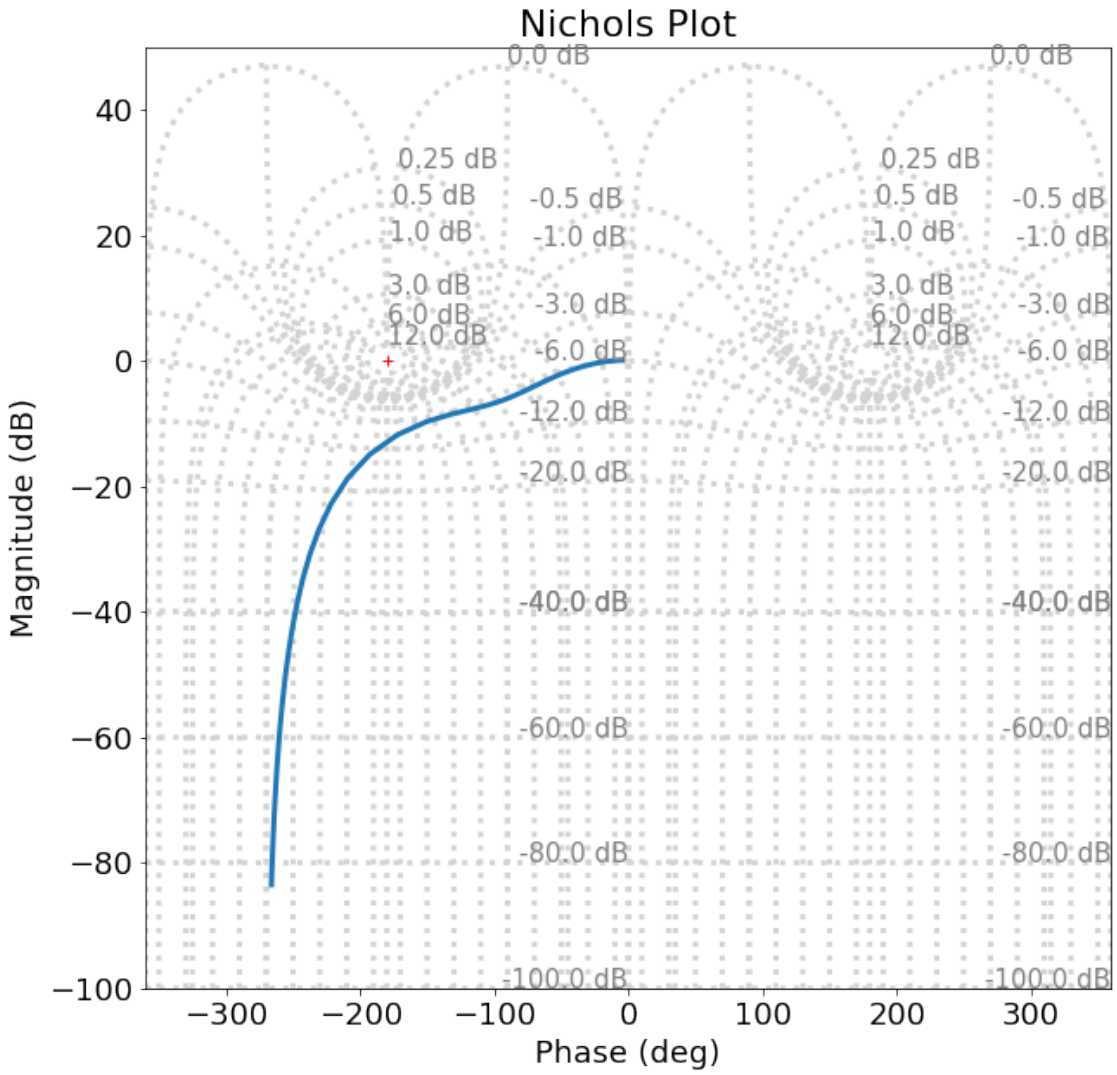


Diagramme de Black-Nichols

[5]:
```
plt.figure(2, [10, 10])
ctl.nichols_plot(SYS_TF)
plt.savefig("black_margin_A1.png", dpi=300)
```



Calcul du gain limite et de la période associée

[6]:
```
INFO_SYSTEME = ctl.stability_margins(SYS_TF)

K_C = INFO_SYSTEME[0] # Marge de gain

W_C = INFO_SYSTEME[3] # Pulsation associée à la marge de gain

T_C = (2*np.pi) / W_C
```

1.3.2 Calcul des paramètres

Correcteur PID

[7]:
```
K_P = 0.6*K_C # Gain proportionnel

T_I = 0.5*T_C # Temps d'intégration

T_D = 0.125*T_C # Temps de dérivation
```

[8]:
```
##############################################################################
# Correcteur PID
#

NUM_PID = [K_P*T_I*T_D, K_P*T_I, K_P*1] # Numérateur de la fonction de transfert

# Dénominateur de la fonction de transfert par ordre décroissant en p ou s
DENUM_PID = [T_I, 0]

SYS_PID = ctl.tf(NUM_PID, DENUM_PID) # Fonction de transfert en p
```

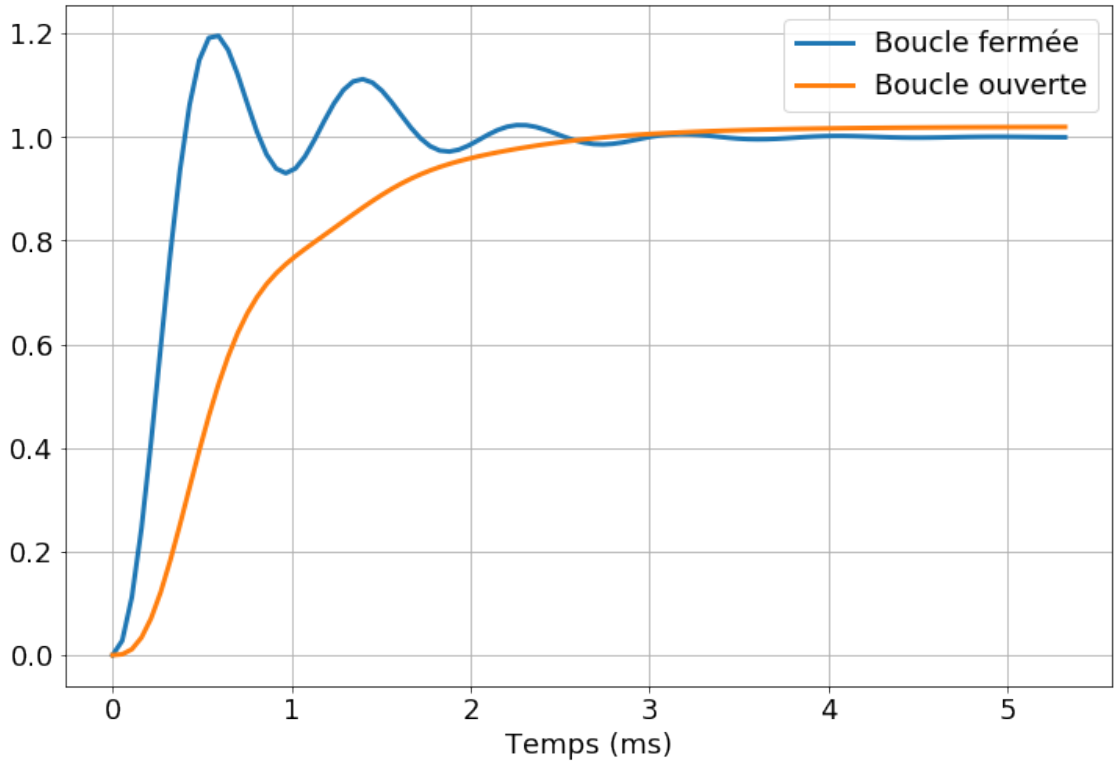
Bouclage du système

[9]:
```
SYS_BF = ctl.feedback(SYS_TF*SYS_PID)
```

Représentation de la réponse indicielle

[10]:
```
t, y = ctl.step_response(SYS_BF) # Calcul de la réponse indicielle en boucle␣
↪fermée
t, y_bo = ctl.step_response(SYS_TF, t) # Calcul de la réponse indicielle du␣
↪système initial
```

[11]:
```
plt.figure(3, [12, 8])
plt.plot(t*1000, y, label='Boucle fermée')
plt.plot(t*1000, y_bo, label='Boucle ouverte')
plt.grid(True)
plt.legend()
plt.xlabel('Temps (ms)')
plt.savefig("rep_boucle_ferm_A1.png", dpi=300)
```


Boucle fermée
Boucle ouverte
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
1
2
3
4
5
Temps (ms)

[ ]: